Generating Composite Overlapping Grids on CAD Geometries

William D. Henshaw

Centre for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, California, 94551 USA
henshaw1@llnl.gov

Abstract

We describe some algorithms and tools that have been developed to generate composite overlapping grids on geometries that have been defined with computer aided design (CAD) programs. This process consists of five main steps. Starting from a description of the surfaces defining the computational domain we (1) correct errors in the CAD representation, (2) determine the topology of the patched-surface, (3) build a global triangulation of the surface, (4) construct structured surface and volume grids using hyperbolic grid generation, and (5) generate the overlapping grid by determining the holes and the interpolation points. The overlapping grid generator which is used for the final step also supports the rapid generation of grids for block-structured adaptive mesh refinement and for moving grids. These algorithms have been implemented as part of the **Overture** object-oriented framework.

Introduction

We briefly describe some algorithms and tools that have been developed to generate overlapping grids on geometries that have been defined with computer aided design (CAD) programs. The fundamental steps in this process are

CAD fixup : When the CAD geometry is defined from a file format, such as IGES, it is usually necessary to repair mistakes in the representation.

CAD connectivity : Since an IGES file usually contains no topology information it is necessary to determine how the trimmed surface patches connect to one another. We construct a global triangulation through an edge matching algorithm.

Surface and Volume Grid Generation : We typically use a hyperbolic grid generator to build both the structured surface grids and the volume grids.

Overlapping Grid Generation : We have developed a new overlapping grid generator, **Ogen** , that determines the overlapping grid (hole cutting, interpolation information) from a collection of overlapping grids.

These algorithms have been implemented as part of the **Overture** object-oriented framework. **Overture** is a toolkit that can be used to develop PDE solvers on overlapping grids and includes the grid generation tools that we describe here. The **OverBlown** Navier-Stokes flow solver which is build using Overture is also available from this web page. **OverBlown** can be used to solve the time-dependent incompressible and compressible Navier-Stokes equations and supports moving grids and adaptive mesh refinement. **Overture** and **OverBlown** are freely available from `www.llnl.gov/CASC/Overture`.

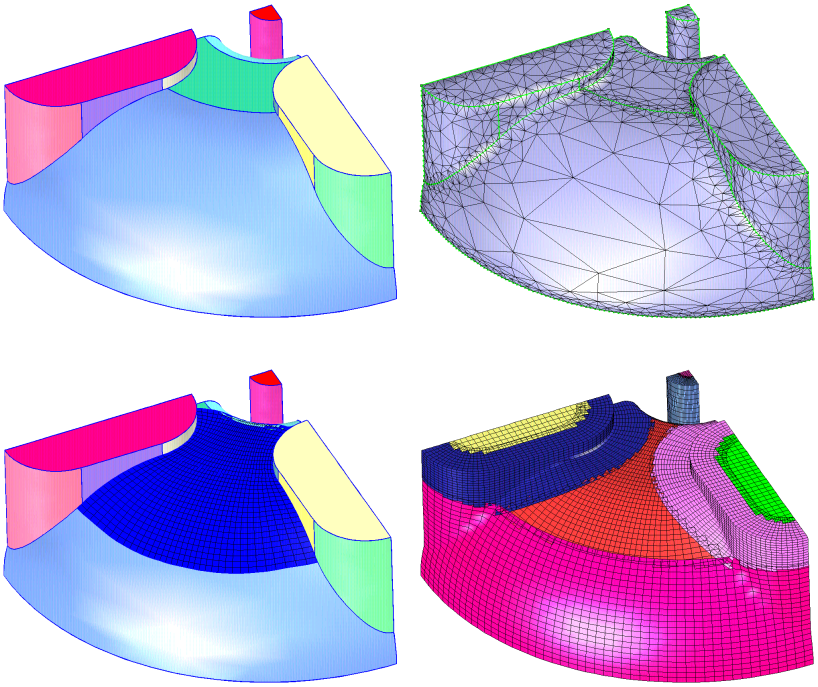

Figure 1. Top left: a CAD geometry represented as a composite surface of multiple trimmed surfaces. Top right: global triangulation for the geometry. Bottom left: a surface grid is generated with the hyperbolic grid generator. Bottom right: overlapping grid for the geometry.

Preparation of the CAD geometry

The geometry is usually defined as a patched-surface consisting of a set of sub-surfaces (or patches), see for example figure (1). A sub-surface may be defined in a variety of ways such as with a spline, B-spline or non-uniform-rational-b-spline (NURBS). In general the sub-surface will be trimmed, in which case only a portion of the surface will be used, the valid region is defined by trimming curves, see figure (2). The output from a CAD program will often be saved in a standard file format such as IGES or the newer STEP specification. Unfortunately the typical IGES output file does not include any connectivity (topology) information; that is there is no information specifying how a given patch connects to other neighbouring patches. To further complicate matters the trimmed patches will often be inaccurate, or contain mistakes, making it difficult to determine where two neighbouring patches should be joined.

CAD fixup: Errors in the geometry representation are fixed in stages. Gross errors in the trimming curves are detected when the geometry is first read from the database file. Errors detected at this initial step include trim curves that lie outside the unit square in parameter space, trim curves that don't close on themselves (i.e. they should be periodic), and trim curves that self-intersect. These gross errors should be fixed before proceeding to the connectivity stage. In Overture we have the ability to edit the trim curves to fix these types of errors. See Petersson and Chand [13] for more details of how we repair CAD geometries. Errors detected later at the connectivity stage would include large gaps between patches or multiple definition of patches. There are many approaches to fixing CAD and removing unwanted details, see for example [10][14].

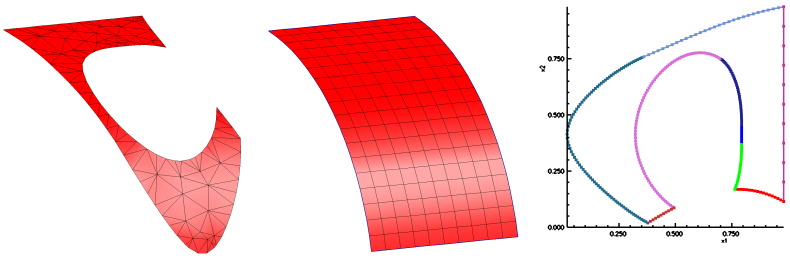

Figure 2. The trimmed patch (top) is formed from an untrimmed surface (middle) and a set of one or more trimming curves (bottom). The untrimmed surface is a mapping from two-dimensional parameter space into three-dimensional cartesian space. The trimming curves are defined in parameter space. The trimming curve is defined in the CAD file as a set of sub-curves (bottom).

CAD connectivity: The topology of the CAD geometry is determined using an *edge-curve* matching algorithm whereby the boundary edges of trimmed surfaces are merged with the boundary edges of neighbouring patches. Our algorithm is similar to that of [17] but the idea has been used elsewhere [10][14].

This technique, shown in figure (3) begins by building curves (edge-curves) on the boundaries of all trimmed-patches and then attempts to identify where an edge-curve from one patch matches to the edge-curve of a neighbouring patch. It is usually necessary to split the edge-curves at appropriate locations in order to perform the matching. When two edge-curves are identified to be the same we say the edges have been merged and choose one edge-curve to define the boundary segment for both patches. By merging edge-curves we effectively remove any gaps or overlaps present in the original representation. See [7] for further details.

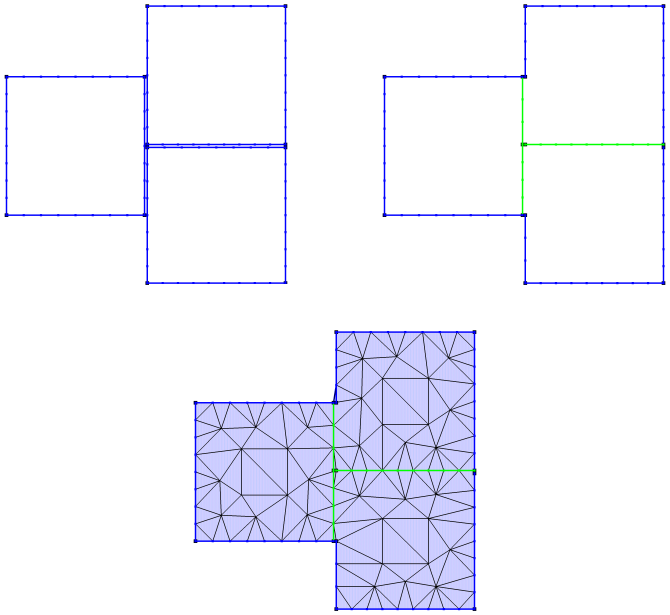

Figure 3. Figure showing the three stages of determining the connectivity. (1) Edge-curves are built on each side of each patch. (2) The edge-curves are merged and then split and merged. (3) Triangulations are built separately for each patch and then stitched together at the common boundary points (bottom).

Global triangulation: A global triangulation can be built once the edge-curves

have been merged. When two edge-curves are merged, one of the two curves is defined to be the true edge-curve. The trimmed-patches are redefined to use the common edge. The global triangulation is formed by initially triangulating each surface patch independently. The triangulation for a patch uses pre-defined nodes on the edge-curves that form the boundary of the trimmed-patch and thus triangulations for adjacent patches will share common nodes. The surface patches are triangulated in the parameter space of the patch. This allows us to use fast two-dimensional triangulation algorithms. We use the *triangle* program from Shewchuk [15] to compute a constrained Delaunay triangulation. After the patch has been triangulated in parameter space it is a simple matter to map the 2D parameter space nodes to 3D. The triangulations for the patches must be stitched together to form a global triangulation. Figure (1) shows a global triangulation built in this way.

Projection algorithm: The global triangulation serves as a basis for a fast algorithm for projecting points onto the patched surface. This projection algorithm is used by the hyperbolic surface grid generator. To project a point onto the patched surface we start by projecting the point onto the global triangulation. Finding the closest triangle is performed by a walking-algorithm if an initial guess is known or otherwise by a global search using an alternating-digital-tree (ADT) tree. Since each triangle belongs to just one sub-patch we can then project the point onto the sub-patch using Newton's method.

Surface and Volume Grid Generation

Structured surface grids can be generated using hyperbolic grid generation. This approach was developed by Steger, Chan and Buning [3, 1, 2] and is also available in Gridgen, see Steinbrenner and Chawner [16]. We have implemented our own version within the Overture framework [6].

The basic marching equations to determine the surface grid $\mathbf{x}(r,t)$, on a surface $\mathbf{C}(\mathbf{x}) = 0$, are defined by the hyperbolic PDE

$$\begin{aligned}
\mathbf{x}_t &= S(r,t)\,\mathbf{n}(r,t) \\
\mathbf{x}(r,0) &= \mathbf{x}_0(r)\ , \text{ initial curve} \\
\mathbf{C}(\mathbf{x}(r,t)) &= 0\ , \text{ grid is constrained to } \mathbf{C}(\mathbf{x}) = 0 \\
B(\mathbf{x}(r,t)) &= 0\ , \text{ boundary conditions}
\end{aligned}$$

where

$$\mathbf{n}(r,t) = \frac{\mathbf{x}_r \times \mathbf{n}_s}{\|\mathbf{x}_r \times \mathbf{n}_s\|} \ , \text{ normal to the front,}$$
$$\mathbf{n}_s \ : \text{ normal to the surface } \mathbf{C} \text{ at } \mathbf{x},$$
$$S(r,t) \ : \text{ scalar speed function.}$$

There are a variety of ways to define the speed function S. See [6] for some possible approaches. These equations march the grid in the direction locally orthogonal to the current front. The parameter t is a time like variable.

Figure (1) shows a surface grid generated on a geometry. Volume grids can also be generated with hyperbolic grid generation. The algorithm is basically the same as that described above.

Overlapping Grid Generation

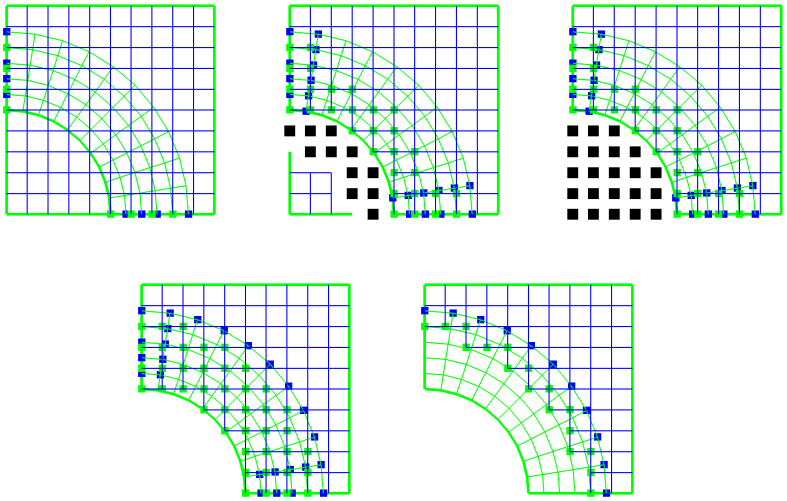

Figure 4. Steps in the overlapping grid algorithm, left to right, top to bottom. 1) After interpolating boundaries of grids that share a edge with another grid (interpolation points are shown small squares). 2) After cutting holes. Each physical boundary cuts hole points (large squares) in other grids. 3) The hole points are swept out starting from the holes computed in the previous step. 4) All the possible interpolation points are found so that excess overlap can removed or so that better quality interpolation points can be used. 5) The final overlapping grid. The excess overlap has been reduced to the (user specified) amount.

The overlapping grid generation algorithm determines how the different component grids communicate with each other. The algorithm must also determine those parts of component grids that are removed from the computation because that part of the grid either lies underneath another grid of higher priority or else that part of the grid lies outside the domain. The determination of the *hole region* is the critical and key step of the algorithm. Other overlapping grid generators include PEGSUS [18], Beggar [8], DCF[9] and xCog/Chalmesh [11][12].

We have developed a new overlapping grid generator called **Ogen** [5]. **Ogen** is based on the CMPGRD grid generator [4] but usually a substantially different algorithm resulting in a more robust approach. Some of the key features of **Ogen** are

1. new hole cutting algorithm with a graceful failure mode.
2. corrections for boundaries of grids that overlap but do not match; this is an especially troublesome problem when the grids are highly stretched.
3. support for higher order discretizations and interpolation.
4. support for cell-centred and vertex-centered interpolation.
5. optimized algorithms for moving grids.
6. support for block structured adaptive mesh refinement.

The basic steps in the overlapping grid algorithm are shown in figure (4). Figure (5) shows an example of a moving grid computation and an adaptive mesh refinement computation. In these examples the grid generator must be called repeatedly to regenerate the grids. Optimized versions of the overlap algorithm are used for this purpose. Further details will appear in a future paper.

Acknowledgments: Thanks to the many people who have contributed to **Overture** in particular the current members of the **Overture** team, David Brown, Kyle Chand, Petri Fast, Brian Miller, Dan Quinlan, Bobby Phillip, Anders Petersson and Don Schwendeman. This research was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

[1] W. CHAN AND P. BUNING, *A hyperbolic surface grid generation scheme and its applications*, paper 94-2208, AIAA, 1994.

[2] W. M. CHAN, *Hyperbolic methods for surface and field grid generation*, in Handbook of Grid Generation, J. F. Thompson, B. K. Soni, and N. P. Weatherill, eds., CRC Press, 1999, ch. 5, pp. 1–26.

[3] W. M. CHAN AND J. L. STEGER, *Enhancements of a three-dimensional hyperbolic grid generation scheme*, Applied Mathematics and Computation, 51 (1992), pp. 181–205.

[4] G. CHESSHIRE AND W. HENSHAW, *Composite overlapping meshes for the solution of partial differential equations*, J. Comp. Phys., 90 (1990), pp. 1–64.

[5] W. HENSHAW, *Ogen: An overlapping grid generator for Overture*, Research Report UCRL-MA-132237, Lawrence Livermore National Laboratory, 1998.

[6] ——, *The Overture hyperbolic grid generator, user guide, version 1.0*, Research Report UCRL-MA-134240, Lawrence Livermore National Laboratory, 1999.

[7] W. D. HENSHAW, *An algorithm for projecting points onto a patched CAD model*, Research Report UCRL-JC-144016, Lawrence Livermore National Laboratory, 2001. Submitted for publication.

[8] R. MAPLE AND D. BELK, *A new approach to domain decomposition, the beggar code*, in Numerical Grid Genertation in Computational Fluid Dynamics and Related Fields, N. Weatherill, ed., Pineridge Press Limited, 1994, pp. 305–314.

[9] R. MEAKIN, *A new method for establishing intergrid communication among systems of overset grids*, AIAA paper 91-1586-CP, American Institute of Aeronautics and Astronautics, 1991.

[10] A. MEZENTSEV AND T. WOEHLER, *Methods and algorithms of automated CAD repair for incremental surface meshing*, in 8th International Meshing Roundtable, 1999, pp. 299–309.

[11] N. A. PETERSSON, *An algorithm for assembling overlapping grid systems*, SIAM J. Sci. Comp., 20 (1999), pp. 1995–2021.

[12] ——, *Hole-cutting for three-dimensional overlapping grids*, SIAM J. Sci. Comp., 21 (1999), pp. 646–665.

[13] N. A. PETERSSON AND K. K. CHAND, *Detecting translation errors in CAD surfaces and preparing geometries for mesh generation*, in Proceeding of the 10th International Meshing Rountable, 2001.

[14] A. SHEFFER, T. BLACKER, J. CLEMENTS, AND M. BERCOVIER, *Virtual topology operators for meshing*, in 6th International Meshing Roundtable, 1997, pp. 49–66.

[15] J. R. SHEWCHUK, *Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator*, in Applied Computational Geometry: Towards Geometric Engineering, M. C. Lin and D. Manocha, eds., vol. 1148 of Lecture Notes in Computer Science, Springer-Verlag, May 1996, pp. 203–222. From the First ACM Workshop on Applied Computational Geometry.

[16] J. STEINBRENNER AND J. CHAWNER, *Gridgen's implementation of partial differential equation based structured grid generation methods*, in 8th International Meshing Rountable, 1998. www.andrew.cmu.edu/user/sowen/-imr8.html.

[17] J. STEINBRENNER, N. WYMAN, AND J. CHAWNER, *Fast surface meshing on imperfect CAD models*, in 9th International Meshing Rountable, 2000. www.andrew.cmu.edu/user/sowen/imr9.html.

[18] N. SUHS AND R. TRAMEL, *Pegsus 4.0 user's manual*, Research Report AEDC-TR-91-8, Arnold Engineering Development Center, Arnold AFB, TN, 1991.

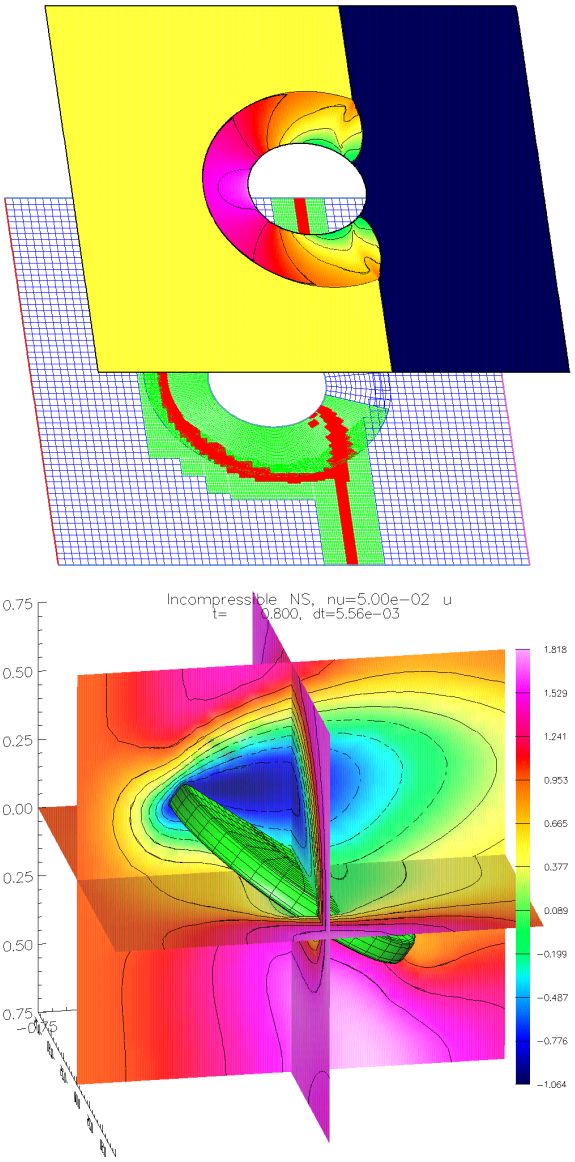

Figure 5. Ogen supports the rapid generation of block-structured adaptive mesh refinement grids and moving grids. The figures show solutions computed by **OverBlown**, of the equations of gas dynamics (top) and incompressible Navier-Stokes equations (bottom).